Preprint
UCRL-JC-145770

Input/Output Scalability of Genomic Alignment: How to Configure a Computational Biology Cluster

P. Vaidyanathan
T. M. Madhyastha
T. R. Jones

U.S. Department of Energy

Lawrence Livermore National Laboratory

This article was submitted to the International Parallel and Distributed Processing Symposium (IPDPS)
Fort Lauderdale, FL
April 15–19, 2002

October 3, 2001

Input/Output Scalability of Genomic Alignment: How to Configure a Computational Biology Cluster

Preethy Vaidyanathan $*$ Tara M. Madhyastha †
$*$ Department of Computer Science
† Department of Computer Engineering

University of California Santa Cruz
1156 High Street
Santa Cruz, CA 95064
{preethy,tara}@soe.ucsc.edu

Terry Jones
Scientific Computing
and Communications Department

Lawrence Livermore National Laboratory
Box 808, L-561
Livermore, CA 94551
trj@llnl.gov

Abstract

Many scientific applications are I/O-intensive, which makes optimization and scaling difficult, especially on parallel architectures. The I/O requirements of computational biology applications are different from other scientific applications. The main difference is that many computational biology applications are embarrassingly parallel and require repeated read-only access to a large global database.

In this paper we examine the scalability of an embarrassingly parallel computational biology application: psLayout, which played a crucial role in the mapping of the human genome. This study was carried out on three architecture: the native UCSC Linux cluster, a Linux cluster at Lawrence Livermore National Labs with a faster interconnect and NFS server, and the ASCI Blue-Pacific supercomputer. We show that a cluster equipped with a fast network and parallel file system or a scalable NFS server has reasonable I/O scalability. We believe that replication is an important issue when scaling to larger numbers of processors, and we introduce the design of a library for automatic data replication to address this issue.

1 Introduction

Computational biology, or bioinformatics, is an extremely important and growing research area. One challenge faced by this field is to understand the makeup of the human genome, revolutionizing our understanding of the human developmental processes and our ability to treat and diagnose diseases. Bioinformatics applications typically have different characteristics than other scientific applications that have enormous data storage and processing needs. Many are extremely data-parallel, making them excellent choices for execution on low-cost clusters, yet their I/O requirements justify a high-performance parallel file system.

As a specific example, consider the input/output requirements of the Human Genome project, the goal of which is to discover all the approximate 30,000 human genes (the human genome) and sequence the 3 billion chemical base pairs making up the human genome [7]. The current size of the genomic database, which is doubling every 14 months, is 40.7 GB [9]. In this paper, we examine the I/O scalability of an embarrassingly parallel computational biology application for sequence alignment, psLayout, that played an important role in the mapping of the human genome [14, 20]. This application is responsible for over 50% of the CBSE cluster use.

We compared the performance of this application on three different architectures: the native Center for Biomolecular Science and Engineering (CBSE) Linux cluster; Vivid, a Linux cluster with a faster interconnect and a Network Appliance NFS server; and ASCI Blue-Pacific. We conclude that either a fast interconnect and parallel file system or high performance NFS server are necessary to adequately meet the I/O needs of this application. Extrapolating trends both in bioinformatics and storage, we anticipate that persistent caching of read-only data will become crucial to I/O performance.

The remainder of this paper is organized as follows. In §2, we present related work. We present trends in the areas of storage and computational biology in §3. We describe computational biology activity on three different architectures in §4. We characterize the behavior of psLayout, the most I/O intensive program being run on the CBSE cluster, in §5. We briefly describe the design of an I/O library for automatic storage replication, motivated by this study, in §6. Finally, we conclude with directions for future work in §7.

2 Related Work

Many researchers have studied the I/O behavior of important high-performance applications out of growing concern over the increasing gap between I/O and processor performance. The CHARISMA project [25] has examined system-level input/output accesses on the iPSC/860 Concurrent File System (CFS) and the CM5 Scalable Disk Array to obtain some generalizations of access patterns in production parallel input/output workloads. They have observed predominantly write accesses, small request sizes, and generally sequential requests. Researchers have characterized the application level behavior of a wide variety of parallel applications [33], and identified difficulties with obtaining high performance from general I/O application interfaces, leading to the development of MPI-IO [24]. Some example application areas from these efforts include modeling of electron-molecule collisions, a 3-D numerical simulation of the Navier-Stokes equations, an implementation of the Hartree-Fock self consistent field method to calculate the electron density around a molecule, and quantum chemical reaction dynamics [15, 31, 32].

These characterization efforts revealed I/O to be a significant component of execution time, but they did not focus specifically on computational biology. A study of the NWS gene sequencing algorithm [11] showed that I/O patterns could be described as a work queue, where each process would compute on some portion of data for either a very short or extremely long period of time, depending on the possibility of a match. Yap *et al* [22] studied the efficiency of parallel algorithms for homologous sequence searching and multiple sequence alignment, demonstrating the importance of load balancing. A key difference in the I/O access patterns of computational biology applications from other scientific applications is that they are often data parallel and read-intensive [26].

3 Bioinformatics and Storage Industry Trends

Figure 1 shows the growth of the Genbank database [5] and the cost to store it [6, 23] over the last 20 years. During this time, the size of the Genbank database has been approximately doubling every 14 months [9]. Meanwhile, the cost of disk storage has been driven down exponentially

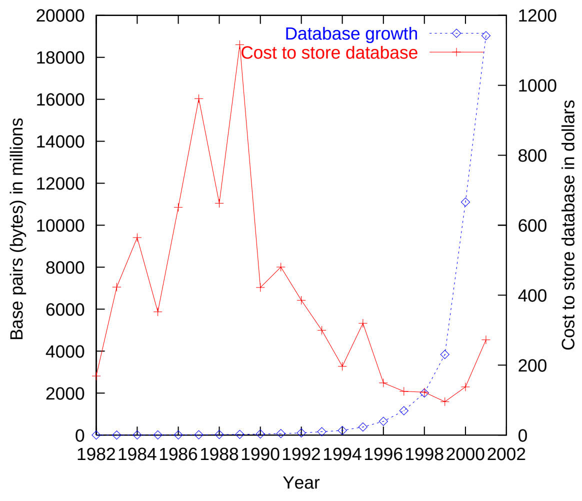


Figure 1: Cost to store the growing genbank database.

primarily by improvements in areal density. The shape of the curve depicting the cost to store the database is therefore governed by the relative rates of database growth and decreasing storage cost.

Current disk drive technology cannot continue this trend without bound. As bits become smaller, the probability that they will spontaneously reverse polarity increases; this is called the superparamagnetic effect. Although the precise density at which this effect will have an impact is unknown, it is motivating a variety of research on alternative storage technologies. IBM predicts current growth will continue for the next four years and then decline [16]. Nevertheless, the volume of bioinformatics data available to researchers has been exploding. Scientists are using hundreds of data formats that are rapidly changing with new technology. We believe that in the next ten years, the storage cost of genomic data will increase, making data management and scalability a serious problem for this class of applications.

4 Computational Biology on Clusters

The bioinformatics group at UCSC is working on several interesting and highly innovative projects; however, the most visible is the mapping of the human genome [14, 7]. The human genome mapping will help us to better understand the human body, biological processes responsible for disease, and differences among species.

Clusters are excellent choices for many bioinformatics problems that are data parallel and do not require high-performance communication. In §4.1 we describe the characteristics of PsLayout, a genomic alignment program with characteristics typical of cluster applications, and in §4.2 we describe several architectures on which we executed this application.

4.1 PsLayout

The basic carrier of genetic information is Deoxyribonucleic acid (DNA), which can be represented as a sequence of nucleotide bases: A-Adenine, C-Cytocine, G-Guanine and T-Thymine. Thus, a DNA molecule is stored as a string over an alphabet of four characters {A,T,G,C} (nucleotides).

```
(a)     maryhadalittlelamblittlelamb

(b)     mary
        hadalittlelamb
        littlelamblittlelamb
        lelamb

(c)     hadalittlelamb
              littlelamblittlelamb
                  lelamb
                              lelamb

(d)     mary
```

Figure 2: Alignment problems: (a) An input containing repeating elements to be aligned with itself (b) Deconstructing the input (c) Assembling overlapping pieces with placement uncertainty (d) Piece does not fit in the assembly.

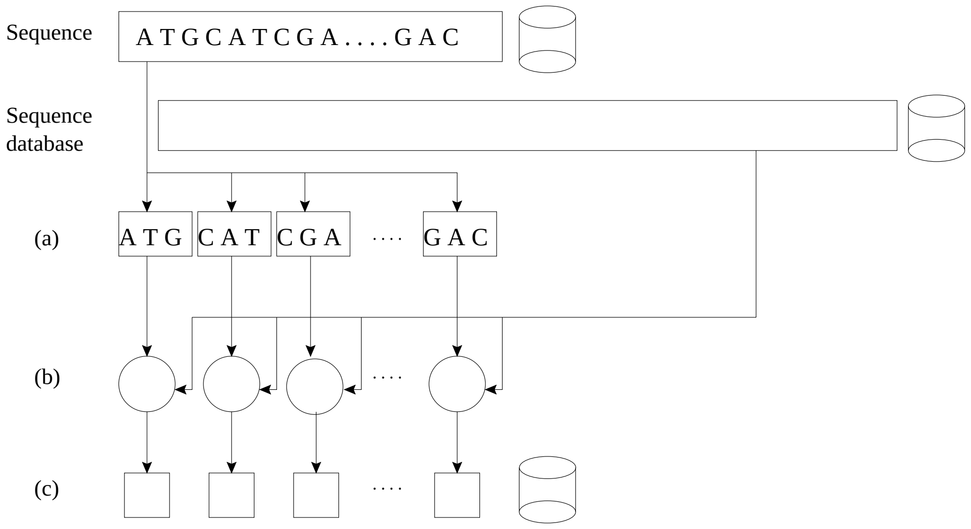


Figure 3: PsLayout execution flow (a) The split of one of the inputs to *n* jobs (sequences) (b) Compute nodes in the cluster aligning sequence to the sequence database, stitching together local sequences using dynamic programming (c) Output local sequences.

To form a draft of the human genome, individual sequence fragments generated from a variety of distributed sources need to be aligned in their positions on a chromosome map and assembled.

Figure 2 illustrates some difficulties in this process [21] using a nursery rhyme as an example. Figure 2a shows an input, or sequence, which contains repeating elements. Our goal is to align this input with itself. Unfortunately, this input comes in subsequence fragments, as shown in Figure 2b; many of these fragments overlap and have repeated subsequences, complicating the alignment process, because the matches could occur at several places (Figure 2c) or the sequence fragment may not align (Figure 2d).

PsLayout is program that finds alignments. It represents the second and most time consuming step of six [20] of the human genome assembly process. PsLayout aligns sequence data that may have “holes” in it with the sequence database. For production runs, this computation takes on the order of three hours, optimized, on the 100-node Pentium III-based CBSE cluster.

PsLayout is an embarrassingly parallel application. The two sequences are given in two input files. The first input, the *sequence data*, is a collection of FASTA files [17] (ASCII files that represent sequences and their descriptions as text strings) containing up to 5 million bases. In this paper the sequence was either either a chromosome sequence or a sequenced BAC (Bacterial Artificial Chromosome) file. The second input, the *sequence database*, is a single FASTA file. This can be either genomic data or mRNA with no restriction on the number of the bases and in this paper it is a sequence of BAC ends.

The input sequence can be split into *n* pieces and be aligned with the sequence database. These *n* individual alignments can be combined to produce the same result as the non-partitioned alignment. Figure 3 illustrates this execution flow.

The input sequence is split into overlapping pieces that are stored in an index. The index also stores where this piece appears in the complete sequence. The sequence database is split into non-overlapping pieces. Each segment of the sequence database is looked up in the index table, and if present in the index it is considered a hit. There is an alignment if the match is above a certain threshold value. If it is not present in the index, it is a miss and is ignored.

Once all the hits are obtained, they must be recombined, which is done using a dynamic program. The hits are projected on the target file when they are 500 bases apart. Thus, the final alignment in this application is obtained by combining the smaller alignments that have been written to individual output files.

4.2 Architectures

As described in §4.1, psLayout is well-suited to cluster execution, but requires a scalable global store for large genomic data files. We would like to configure a cluster to provide good performance for this class of applications at the lowest cost.

To this end, we examine the performance of psLayout on three different architectures that span a wide range of cost and performance: the native CBSE cluster at UCSC; Vivid, a cluster at Lawrence Livermore National Labs (LLNL) with Myrinet interconnect; and ASCI Blue-Pacific (Blue), a high-performance supercomputer at LLNL. We describe the relevant characteristics of these architectures.

4.2.1 CBSE Cluster

The Center for Biomolecular Science and Engineering (CBSE) at UCSC has a cluster with 93 Linux nodes, which is being extended to 1008 nodes. Each node of the 93-node cluster has an 850 MHz Pentium III processor with 256 MB RAM and a 20 GB IDE drive. Two nodes are NFS servers for the cluster. The nodes are internetworked with 100 Base-T Ethernet in two subclusters. Files may be stored either on local Linux file systems on each node, or globally on NFS; there is no parallel file system.

Jobs are scheduled using Condor [2]. Although Condor is designed for computing using collections of distributed resources, as opposed to parallel computing on a homogeneous cluster, many of the computational biology applications lend themselves well to a work-queue programming model.

4.2.2 Vivid Cluster

The Vivid cluster at LLNL is a 33-node Linux cluster connected by Myrinet [8]. Each node has a 800 MHz Pentium III processor and a local SCSI disk. Files may be stored either at the local disk of each node, or globally on a Network Appliance NFS server or on the Parallel Virtual File System (PVFS). The NetApp NFS server incorporates the WAFL (Write Anywhere File Layout), which provides high-performance NFS service [19]. PVFS [29] stripes files among the local disks of the cluster nodes for better performance compared to a NFS server [3, 13].

4.2.3 ASCI Blue-Pacific

ASCI Blue-Pacific (Blue) is a supercomputer at LLNL with 280 nodes, each with four 332 MHz PowerPC604e processors. The nodes are interconnected by a SP2 switch [1]. The file system is General Parallel File System (GPFS). The global store GPFS provides high performance to run parallel applications by striping I/O across multiple disks [4].

ASCI Blue is used to solve a variety of scientific calculations by using parallel applications. Some of these applications are sPPM to solve compressible turbulence problem [10], MPQC to search the existence of polymeric forms of nitrogen [28], JEEP [18], IMPACT, a coupled atmospheric modeling simulation [30] and Ardra to simulate the flux of fusion neutrons that comes out of the Nova laser target chamber [12]. All these applications are characterized as writes mostly with an ability to restart from datasets of intermediate calculations.

5 I/O Characterization

PsLayout has characteristics typical of computational biology alignment codes; it is highly data parallel and there is no communication except through the file system. The psLayout algorithm was designed when the genomic database was 3 GB; however, the database has currently grown by an order of magnitude. As the volume of data increases, I/O-intense applications become increasingly I/O bound, and tuning the algorithm becomes a moving target.

PsLayout has already been highly tuned for execution on the CBSE cluster by Jim Kent as part of the human genome mapping effort; we study its I/O performance on a variety of architectures to learn what cluster software and hardware architecture supports high performance at the lowest cost.

5.1 PsLayout Overview

PsLayout reads the two input sequences and writes results to an output file. PsLayout is structured so that the alignment computation is inextricably interleaved with the I/O. Because each node running the program needs to access the two input files, a scalable shared store (either a parallel file system or network file system) is necessary.

Each FASTA file in the sequence file is read into memory using a single application read call, which uses buffered I/O to read the region between markers one line at a time. Markers are points indicating safe points to split the input. After the input is read, the index table is generated from the sequence.

The sequence database is a list of FASTA files, which are each read one character at a time. The bytes in the sequence database are compared to the index table. This process generates the individual alignments which are recombined using a dynamic programming approach. These results are written to a file.

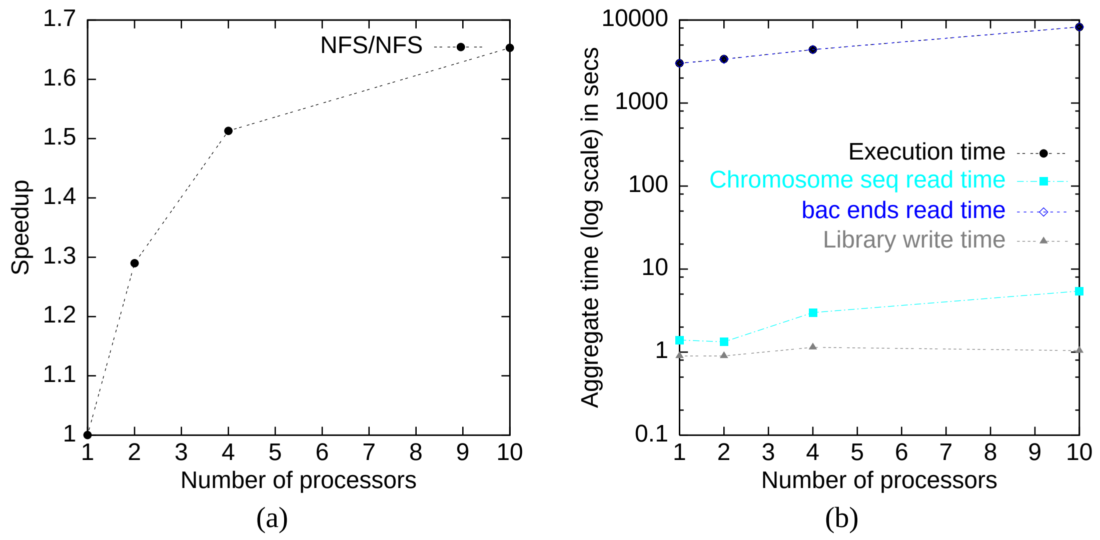


Figure 4: PsLayout run on CBSE cluster with 4 MB and 411 MB input files on NFS servers (a) Speedup (b) Aggregate time breakdown.

Most of the application time (approximately 98-99%) is spent in user-level I/O libraries doing buffered reads, memory allocation, and string comparisons. The write time is insignificant by comparison.

5.2 Application Instrumentation

We instrumented the application-level I/O calls using the Pablo [27] performance environment, which supports user-level performance data capture and analysis. As described in §5.1, reading and alignment are tightly interleaved.

PsLayout runs in parallel as separate applications on different nodes, creating several individual trace files, one for each portion of the job. We combine these files to achieve a global temporal ordering; clocks on the individual nodes are synchronized using NTP. Tracing overhead was negligible.

5.3 Characterization Results

PsLayout is embarrassingly parallel, with no communication between nodes except through I/O, and should ideally scale very well. However, as shown by Figure 4a, it does not. Figure 4a shows speedup of psLayout on the CBSE cluster using very small input data sets (4 MB and 411 MB), both located on the global store. With 10 processors, the speedup is approximately 1.65. For this experiment, the sequence file is a 4 MB chromosome sequence and the 411 MB sequence database is bac ends taken from Bacterial Artificial Chromosome (BAC). This run took about 50 minutes to complete on a single CBSE processor.

In Figure 4b, we examine the breakdown of the execution time for psLayout. Here we compare the aggregate of individual execution times on separate nodes as we scale the number of processors. The difference between aggregate execution and sequence database read time is negligible, indicating that most of the alignment computation is occurring interleaved with the character by

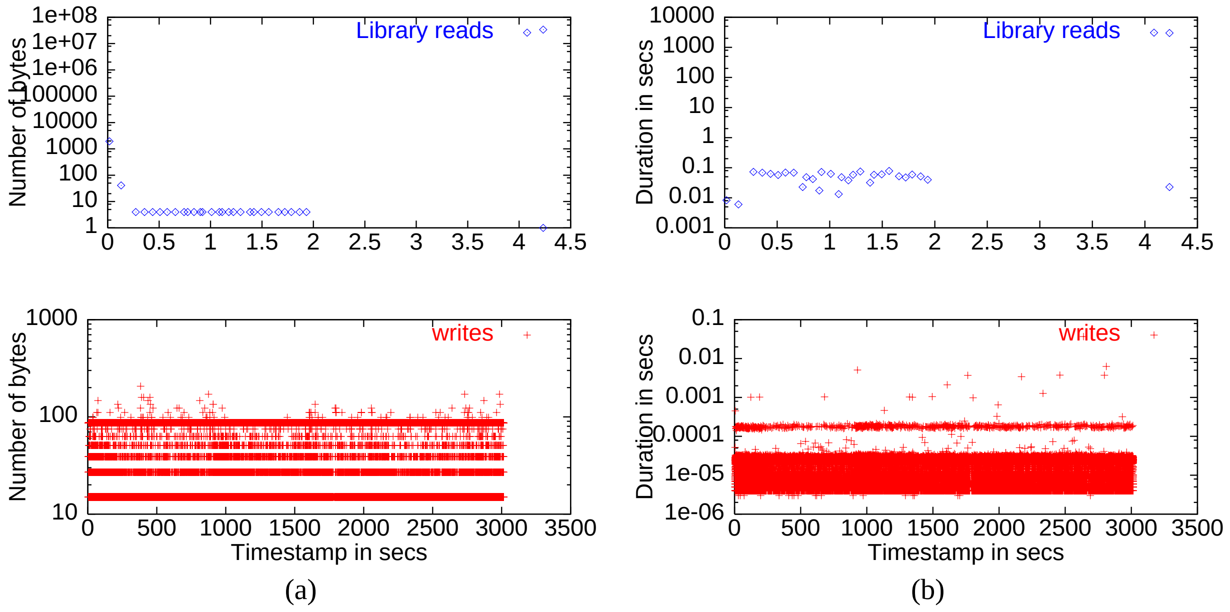


Figure 5: Timeline of psLayout on CBSE cluster for 4 MB and 411 MB NFS file inputs (a) Number of bytes accessed through library reads and writes (b) Duration of these library reads and writes. Note that the vertical axes are in logarithmic scale.

character read of the sequence database. The write time represents a very small fraction of the total execution time (less than 0.05%).

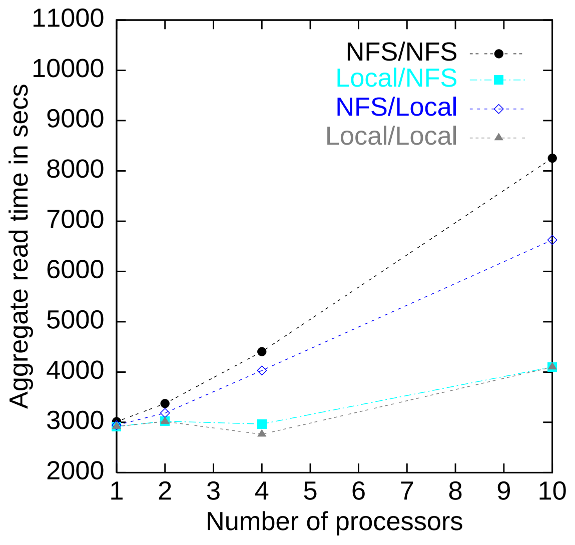
As shown in Figure 5, some large application reads occur in the beginning of execution, followed by many small writes. This confirms our description of I/O activity given in §4.1. Some of the reads are exceptionally long; this variance is caused by the lengths of the sequences and the difficulty of alignment.

5.3.1 Input File Location

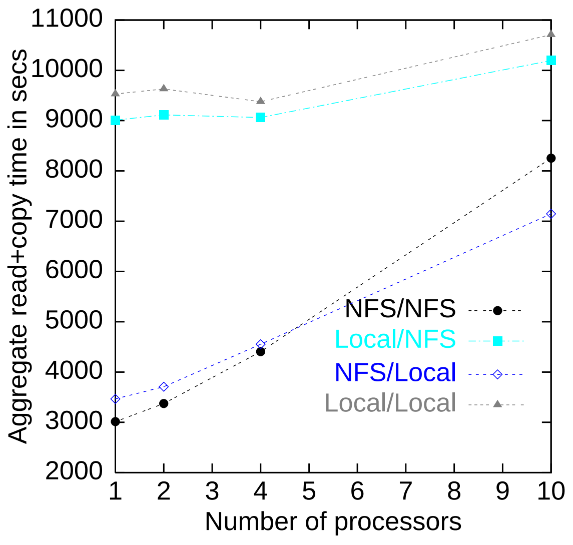
The major suspect in the poor scalability of the CBSE cluster shown in Figure 4a is file location. Files are shared on the CBSE cluster using NFS, but we know this performs poorly under concurrent requests, evident from Figures 5a and 5b. For convenience, databases are kept at the global store; however, for performance reasons, we could consider replicating either one or both of the inputs at the local node disks.

For both applications, the two input files can be at the global store, or one on the global store and the other at local disk, or both on local disk, creating four combinations: global/global, local/global, global/local and local/local. The last three combinations incur copy time for copying one or more input file to the local store. For the CBSE cluster, files must be copied to all nodes, because we do not know initially what node Condor is going to use. We use an optimized binary tree copy program that understands the cluster topology. For Blue and Vivid, we need copy only to those processors doing the alignment.

To understand the impact of file location on performance, we execute PsLayout using the same input files but consider the effect of replicating them from global to local store, adding copy time to the overall execution time. The copy time is significant for the CBSE cluster because of the larger



(a)



(b)

Figure 6: Comparison of read times with increased levels of parallelism of psLayout on CBSE cluster for 4 MB and 411 MB file inputs and different file locations (a) excluding copy time overhead (b) including copy time overhead.

number of processors to which files must be copied (because Condor does not know which nodes will run the jobs) and the slower interconnect. For Blue and Vivid, the copy time is insignificant primarily because of their faster interconnects and secondarily because files are copied to only the specific processors being used. For example, for the 411 MB file, copy time to 100 nodes on the CBSE cluster is 6093 secs, in contrast to 62 secs on Vivid cluster to copy it to one node, and 82 seconds to copy it to 30 nodes, a difference of two orders of magnitude.

Figure 6a and 6b show aggregate application read time for psLayout excluding and including copy overhead, respectively, for different file locations on the CBSE cluster. As described in §5.3, application read calls account for more than 99% of the total execution time and include the work both of reading and aligning. Because there is no overhead caused by splitting the work among many nodes, the curves in Figure 6a and 6b should be close to horizontal. Instead we see in Figure 6a that the local/NFS and NFS/NFS curves begin to increase significantly even at four processors, indicating that NFS is, as predicted, a bottleneck. Depending on the precise split, there are actually slight variances in the total time to perform the alignment; this is why local/local and NFS/local for four processors takes slightly less time than for two processors and slightly more for 10.

Unfortunately, Figure 6b shows that the cost of copying files can erode the performance improvement. We did not run experiments on the CBSE cluster using NFS for global storage for larger experiments with larger numbers of processors because it is obvious that there is a crossover point where the additional copy overhead is insignificant compared to the overhead caused by the NFS bottleneck. As the number of nodes increases, the benefit of replication becomes clear.

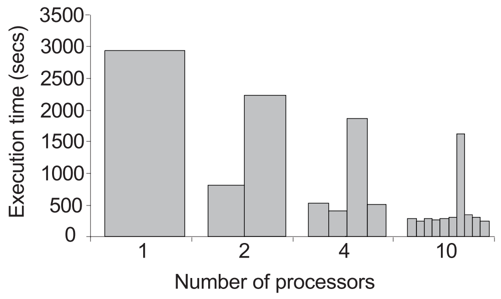


Figure 7: PsLayout workload distribution for 4 MB and 411 MB local file inputs on the CBSE cluster.

5.3.2 Load Balancing

Even when all I/O is local, psLayout does not scale very well. We can see from Figure 7 that this is because of problems with load balancing. There is one node among the 10 processors whose alignment takes at least more than double the average time. The input to this node has a lot of repeats in its sequence, causing delay in the alignment, because several sequences match for each alignment. In practice, scientists manually balance the cluster load by timing the submission of their jobs.

5.3.3 File System Scalability

As shown in §5.3, a global store can be a bottleneck, and the network performance determines to what degree replication can alleviate this bottleneck. Here, we determine how the performance of psLayout scales as we increase the number of processors on different architectures. We used two sets of input files. The first set is the 4 MB and 411 MB dataset described in §5.3, and the second set is a 26 MB sequenced BAC file and the common 411 MB file. We scaled the smaller run up to 10 processors and the larger up to 50 processors.

On the CBSE cluster, large runs are executed using the NFS/local combination to avoid contention. ASCI Blue represents the opposite architectural extreme, where the network infrastructure is fast, and where GPFS is a highly tuned parallel file system available to all nodes. Figure 8 shows the four combinations for ASCI Blue using GPFS and local disk as input file locations. All four combinations perform similarly, and are comparable to the CBSE cluster without considering copy overhead. For the local copy, the file is copied from the NetApp NFS server. With the fast interconnect the copy overhead is insignificant.

Vivid represents a compromise between the CBSE cluster and Blue; it has a fast network and fast NFS server. Figures 9a and 9b show aggregate execution and copy time from Vivid.[1] The possible input file locations are PVFS, NetApp NFS and local disk. For the local disk copy, the file is copied from the NetApp NFS server to the local disk. With a very small copy overhead, both these graphs show good scalability. The NFS/local and PVFS/PVFS combination perform best

[1]Data from the local/local four processor run is unavailable, but will be available for the final version

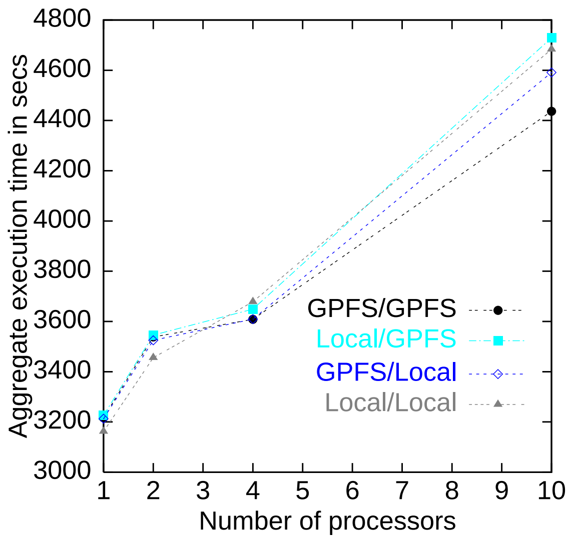


Figure 8: Comparison of execution times with increased levels of parallelism of psLayout on ASCI Blue-Pacific for 4 MB and 411 MB file inputs and different file locations.

when scaling to 10 processors. The PVFS/PVFS input file combination performs slightly better than the NFS/local one for 10 processors.

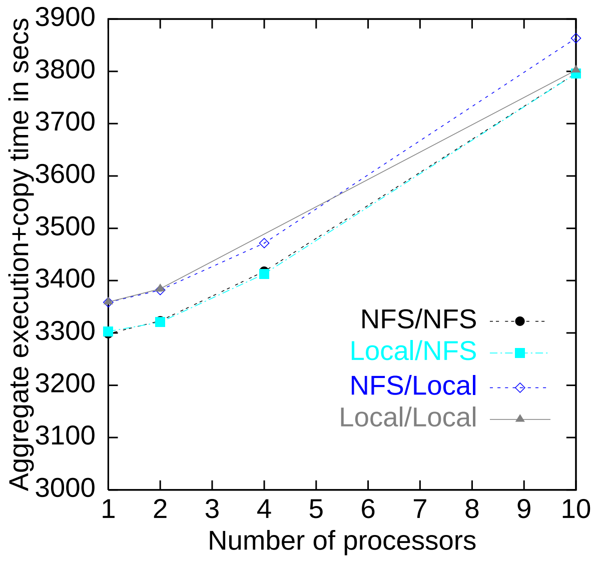
Figure 10a shows the scalability of the 26 MB and the 411 MB input file run. For this larger run, we examined only the NFS/local combination on the CBSE cluster and the NFS/local and NFS/NFS combinations on the Vivid cluster, with the 26 MB file at the NFS server and the 411 MB file at the local disk. The CBSE cluster is a production system, with limited resources and we did not want to risk bringing it down by taxing the NFS server. Vivid has only 33 processors so we cannot scale to 50. The aggregate execution time is relatively constant as the number of processors increases, indicating good scalability. The NFS/NFS combination on Vivid is slower than the NFS/local, although not significantly. The CBSE cluster has better performance than Vivid because of the faster processors on the CBSE cluster.

We examine the scalability of this application on Blue in Figure 10b for different combinations of local and global store. All combinations scale well for 50 processors. For different numbers of processors, different combinations are slightly better or worse.

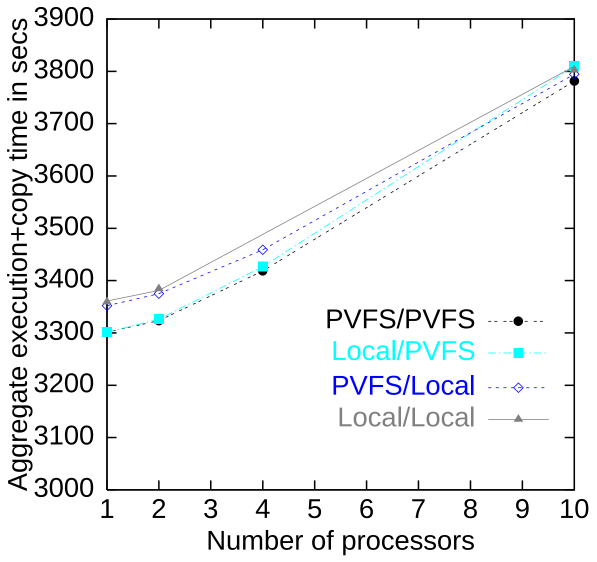
5.4 Summary

PsLayout, a typical computational biology application, has characteristics very different from many scientific applications. It is embarrassingly parallel, with all communication through the shared global store. Scalability of this global store is crucial to performance.

We characterized two experimental runs of psLayout on the three architectures. The best input file location varied based on the number of processors, the size of the input files, and the architecture. A bioinformatics cluster should have either a fast networking infrastructure and a parallel file system (such as PVFS) or access to a scalable NFS server.



(a)



(b)

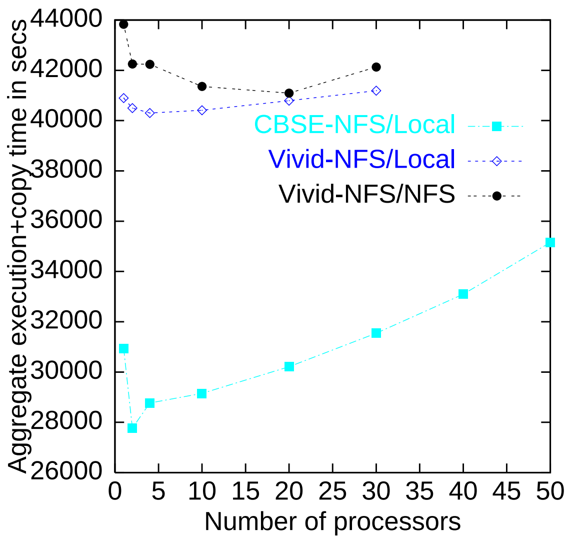
Figure 9: Comparison of execution times with increased levels of parallelism of psLayout on Vivid cluster for 4 MB and 411 MB file inputs and different file locations using (a) NFS server and Local disk (b) PVFS and Local disk.

6 Dynamically Replicated Storage

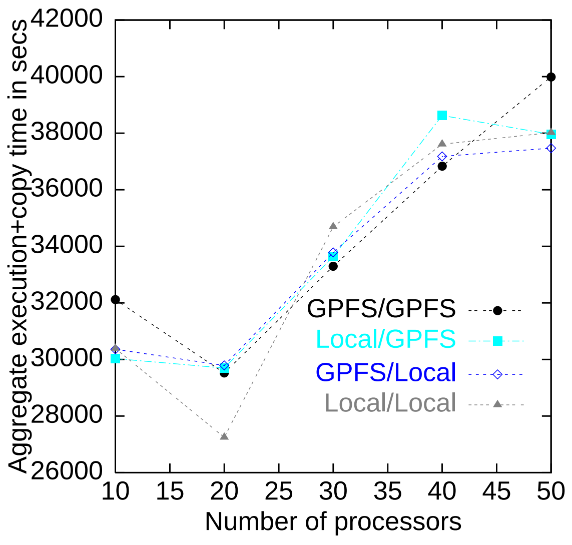
In §5, we showed that I/O scalability problems are evident with even a small degree of parallelism. Programmers at UCSC alleviate bottlenecks by manually replicating databases to improve locality, and this approach works. Given the low cost of storage, replicating databases is a reasonable solution for improving performance but managing these replicas is difficult and time consuming. As explained in §3, storage cost trends and genomic data trends are such that indiscriminate replicas are probably not a cost-effective solution.

To address this problem, we are developing a user-level library for a new model of location-transparent storage. This may be viewed as an extension to existing user-level parallel I/O libraries that not only stripes files, but maintains read-only replicas of records and information about access times. Therefore, a read access to a record may be redirected to the most appropriate location. Unlike a traditional cache, where there is a strict hierarchy of access times (that usually differ by an order of magnitude or more) as shown in Figure 11a, access times to local disk or network storage change based on load and network conditions and may not retain a strict ordering.

Figure 11b shows an example cache table entry for a genomic data file. Here, the cost for accessing data at each location is calculated as a simple function of the number of processors, the file location and the file size. Although these parameters are fixed at the start of application execution, the cost function may be based on parameters that vary continuously. For example, as network links break or bandwidth is limited, it will be more expensive to access a file on the Web, and this can be reflected in this model. Ultimately, we envision linking replication with a dynamic run-time performance model that can provide performance data of the execution environment on-the-fly to calculate access costs.



(a)



(b)

Figure 10: Aggregate execution time of psLayout using 26 MB and 411 MB inputs (a) NFS and local file input on CBSE and Vivid clusters (b) Comparison of execution times for different file locations on ASCI Blue-Pacific.

7 Conclusions and Future Work

Computational biology is an important application area with different I/O needs than other scientific applications. We characterized the performance of psLayout, a computational biology application that performs genomic alignment, on three architectures. We determined that although it is embarrassingly parallel, psLayout has poor scalability due to I/O contention and poor load balancing.

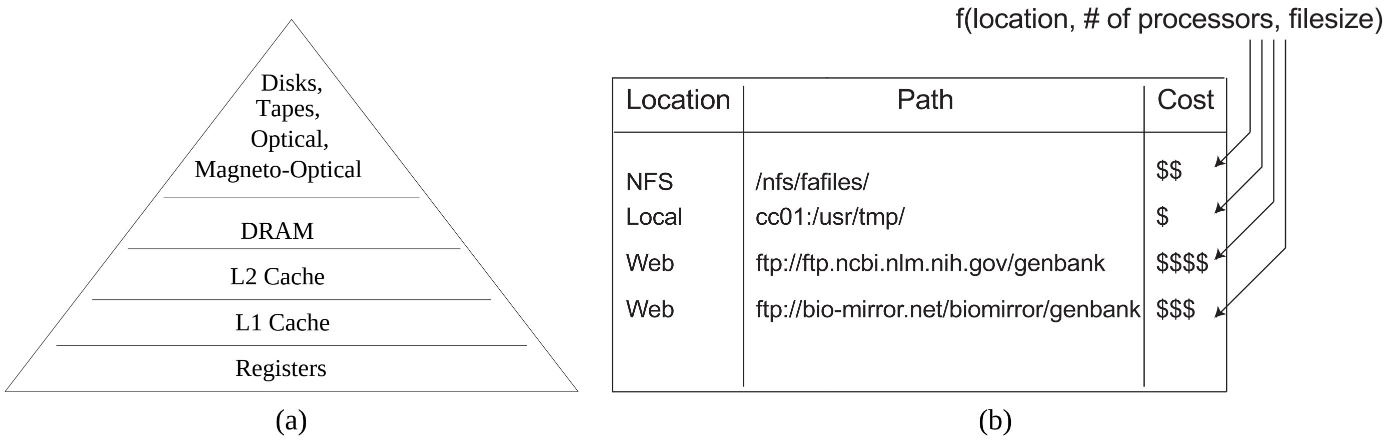
We assessed scalability on a range of file systems and architectures ranging from the low-end CBSE cluster to ASCI-Blue. The best-performing combination of input databases for 10 processors with input file sizes 4 MB and 411 MB is different for each architecture: NFS/Local for CBSE, NFS/Local and PVFS/PVFS for Vivid and GPFS/GPFS for ASCI-Blue. Input file location is a major factor affecting the aggregate execution time of this application.

Although a fast network and parallel file system or a scalable NFS server can service the clusters and loads described in this paper, we believe that replication of data will play an increasing role in scalability of this class of applications. We are currently developing a library to support the data replication that is now performed manually to automatically improve runtime performance.

Acknowledgments

We thank David Haussler for allowing us to use the cluster for our experiments, and we are grateful to Jim Kent, Patrick Gavin and Jorge Garcia for their constant help in answering our questions. This research was partially funded by NSF Grant No. CCR-0093051 and Lawrence Livermore National Labs.

This work was performed under the auspices of the United States Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.



Location	Path	Cost
NFS	/nfs/fafiles/	$$
Local	cc01:/usr/tmp/	$
Web	ftp://ftp.ncbi.nlm.nih.gov/genbank	$$$$
Web	ftp://bio-mirror.net/biomirror/genbank	$$$

Figure 11: (a) Memory Hierarchy (b) Example for cache table entry.

References

[1] ASCI Blue-Pacific.
http://www.llnl.gov/asci/platforms/bluepac/bonus_links.html, 14th Sept 2001.

[2] Condor High Throughput computing. http://www.cs.wisc.edu/condor/, 29th Mar 2001.

[3] File Systems for Clusters from a Protocol Perspective.
http://www.extremelinux.org/activities/usenix99/docs/braam/bra- am.html, 16th Sept 2001.

[4] General Parallel File System for AIX. http://www-1.ibm.com/servers/eserver/pseries/software/sp/gpfs.html, 16th Sept 2001.

[5] Growth of Genbank. http://www.ncbi.nlm.nih.gov/Genbank/genbankstats- .html, 20th July 2001.

[6] Historic notes about the cost of hard drive storage space. {http://www.alts.net/ns1625/winchest.html, 20th July 2001.

[7] Human Genome Project Information. http://www.ornl.gov/hgmis/, 9th Apr 2001.

[8] Myrinet Overview. http://www.myri.com/myrinet/overview/, 14th Sept 2001.

[9] NCBI Databases. http://www.ncbi.nlm.nih.gov:80/Database/index.html, Feb. 2001.

[10] A.A.MIRIN, R.H.COHEN, B.C.CURTIS, W.P.DANNEVIK, A.M.DIMITS, M.A.DUCHAINEAU, D.E.ELIASON, D.R.SCHIKORE, S.E.ANDERSON, D.H.PORTER, P.R.WOODWARD, L.J.SHIEH, AND S.W.WHITE. Very High Resolution Simulation of Compressible Turbulence. In *Supercomputing 99 conference* (1999), no. UCRL-JC-134237.

[11] ARENDT, J. W. Parallel genome sequence comparison using a concurrent file system. Tech. Rep. UIUCDCS-R-91-1674, University of Illinois at Urbana-Champaign, 1991.

[12] BROWN, PETER, CHANG, B., GRANT, K., HANEBUTTE, U. R., S.WOODWARD, C., AND A.BRUNNER, T. ARDRA:Scalable Parallel Code System to Perform Neutron and Radiation Transport Calculations. In *SC99* (1999).

[13] CARNS, P. H., AND {III, W. B. PVFS:A Parallel File System for Linux Clusters.
http://parlweb.parl.clemson.edu/pvfs/el2000/extreme2000.html#- simitci:framework, 14th Sept 2001. Parallel Architecture Research Laboratory.

[14] CONSORTIUM, I. H. G. S. Initial sequencing and analysis of the human genome. *Nature* (Feb. 2001), 860–921.

[15] CRANDALL, P. E., AYDT, R. A., CHIEN, A. A., AND REED, D. A. Characterization of a Suite of Input/Output Intensive Applications. In *Proceedings of Supercomputing '95* (Dec. 1995).

[16] D.A.THOMPSON, AND J.S.BEST. The future of magnetic data storage technology. http://www.research.ibm.com/journal/rd/443/thompson.html, 2000. Volume 44,Number 3.

[17] DR.M.HILL. The Human Genome- About FASTA files. http://anatomy.med.unsw.edu.au/cbl/GENOME/about/aboutfasta.htm, 29th Mar 2001.

[18] FATTEBERT, LUC, J., AND GYGI, F. A continuum solvation model for ab initio molecular dynamics. In *American physical society* (2001).

[19] HITZ, D., LAU, J., AND MALCOLM, M. File system design for an NFS File Server Appliance. Tech. Rep. TR3002, Network Appliance Inc.

[20] KENT, W. Gigassembler: An algorithm for the initial assembly of the human genome working draft, http://genome.ucsc.edu/goldenPath/algo.html. Tech. rep., School of Engineering,University of California,Santa Cruz, 2000.

[21] KENT, W. The Human Genome Project and UCSC. http://www.soe.ucsc.edu/ kent/presentations/ScholarsDay2001/, 28th Sept 2001.

[22] K.YAP, T., FRIEDER, O., AND L.MARTINO, R. Parallel computation in biological sequence analysis. *IEEE Transactions on Parallel and Distributed Systems 9*, 3 (March 1998), 283–294.

[23] L.HENNESSY, J., AND A.PATTERSON, D. *Compter Architecture a quantitative approach*. Morgan Kaufmann Publishers, Inc., 1996.

[24] MPI-IO: a parallel file I/O interface for MPI. The MPI-IO Committee, April 1996. Version 0.5.

[25] NIEUWEJAAR, N., KOTZ, D., PURAKAYASTHA, A., ELLIS, C. S., AND BEST, M. File-access characteristics of parallel scientific workloads. *IEEE Transactions on Parallel and Distributed Systems 7*, 10 (October 1996), 1075–1089.

[26] Personal communication. Thomas R. Slezak, Lawrence Livermore National Labs, September 2001.

[27] REED, D. A., AYDT, R. A., NOE, R. J., ROTH, P. C., SHIELDS, K. A., SCHWARTZ, B., AND TAVERA, L. F. Scalable performance analysis: The pablo performance analysis environment. In *Proceedings of the Scalable Parallel Libraries Conference* (Oct. 1993), IEEE Computer, pp. 104–113.

[28] R.MANAA. Towards a new energy-rich molecular systems: from N10 to N60. In *Chemistry Physics letters* (2000).

[29] ROSS, R. The Parallel Virtual File System.
http://parlweb.parl.clemson.edu/pvfs/, 14th Sept 2001.

[30] ROTMAN, C.ATHERTON, D.BERGMANN, P.CAMERON-SMITH, C.CHUANG, P.CONNELL, J.DIGNON, A.FRANZ, K.GRANT, A.MIRIN, C.MOLENKAMP, AND J.TANNAHILL. IMPACT, A coupled tropospheric/stratospheric chemistry model:Analysis and comparison of results to observations. In *American Geophysical Union Annual Fall meeting* (2000).

[31] SMIRNI, E., AYDT, R. A., CHIEN, A. A., AND REED, D. A. I/O Requirements of Scientific Applications: An Evolutionary View. In *Fifth International Symposium on High Performance Distributed Computing* (1996), pp. 49–59.

[32] SMIRNI, E., AND REED, D. A. Workload characterization of input/output intensive parallel applications. In *Modelling Techniques and Tools for Computer Performance Evaluation* (June 1997).

[33] THAKUR, R., LUSK, E., AND GROPP, W. I/O in parallel applications: The weakest link. *The International Journal of High Performance Computing Applications 12*, 4 (Winter 1998), 389–395.

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.